Roll Call
Documentación de Análisis

Miguel Gómez Casado
(a.k.a. Crazy Jäger)

[logos]

Resumen

Este documento surge como análisis de un proyecto software independiente con el objetivo secundario de proporcionar una muestra física y palpable de mi forma de trabajar. La idea del proyecto en sí surge como resultado de la necesidad de crear una aplicación Android que permita pasar lista a un grupo de personas pertenecientes a un grupo para así poder llevar un control de asistencia de una forma cómoda y sencilla.

Este documento busca proporcionar una documentación formal a un proyecto sencillo y pequeño que defina claramente el alcance funcional del proyecto, así como su estructura y forma de abordar el proyecto. Con ello en mente, tras una breve introducción, se procederá a presentar los requisitos del proyecto, los modelos del aplicativo resultado del análisis, los casos de uso, y la arquitectura y diseño finales del sistema.

Por último, hay que mencionar que este proyecto sigue una estrategia de desarrollo incremental, por lo que los contenidos de este documento pueden verse modificados según surjan nuevas especificaciones y modificaciones a realizar, sobre todo en fases iniciales al proyecto, que afectaras especialmente a los últimos apartados del documento.

Contenido

1. Trasfondo y solución propuesta

Durante mucho tiempo, he practicado judo. Al llegar al cinturón negro, se abrió un nuevo camino donde, además de seguir aprendiendo como alumno, podía impartir clases como profesor a otros que buscasen adentrarse en este deporte. Al principio, los grupos a los que impartía clases eran muy reducidos, con no más de 10 personas, pero a medida que ganaba experiencia y se me abría la posibilidad de impartir clases en pabellones más grandes, el número de alumnos también incrementaba. Este año, junto con otro compañero, empezamos a impartir clases a un grupo de tamaño considerable. Hasta ahora, una hoja de papel o una tabla en una hoja de cálculo era suficiente para llevar el control de asistencia del alumnado, pero, ante un grupo de estas dimensiones, la utilización de una hoja de cálculo se volvía bastante más aparatosa. A este problema se le suma el hecho de la necesidad de reducir objetos materiales tangibles, como el papel y el bolígrafo, debido a la Covid-19, motivo por el cual llevar un estricto control de asistencia es también una prioridad en toda actividad que tenga un número considerable de participantes, especialmente en un deporte de contacto. A raíz de todo esto, aparece la idea de crear una aplicación móvil que facilite pasar lista a los asistentes.

Inicialmente, este era solo un pequeño proyecto personal que tenía como objetivo facilitar el desarrollo de una actividad que realizaba con cierta regularidad. A consecuencia de ello, iba a tratarse de un pequeño desarrollo con baja calidad y sin ningún esfuerzo añadido en hacer una documentación formal como ésta. A medida que pasaba el tiempo, sin embargo, se hizo evidente que este proyecto abría la posibilidad de hacer pública mi forma de operar en un aspecto de la ingeniería de software que muchos compañeros de la profesión informática encuentran desagradable e incluso tedioso; pero que, como en cualquier proceso de ingeniería, puede llegar a ser de vital importancia para un proyecto, especialmente si dicho proyecto tiene un tamaño considerable o va a pasar por muchas manos durante su desarrollo y mantenimiento. La percatación de este hecho sería lo que marcaría el punto de inicio del desarrollo de esta documentación y, posteriormente, el de la aplicación.

El objetivo motivaba la realización del proyecto, y la posibilidad de mostrar mis habilidades en este campo motivaban la realización de la documentación de éste, así como su publicación. Con la decisión tomada de hacer el proyecto público, se abría una última puerta que había querido explorar desde hace ya algún tiempo: los derechos de autor y las licencias.

Conozco, gracias a amigos y familiares, historias de informáticos que, con una motivación e ilusión como la mía, habían desarrollado por cuenta propia un software que les era después arrebatado, a menudo por el propio cliente, quien se atribuía a sí mismo la autoría del trabajo y lo explotaba sin ningún respeto hacia quienes habían invertido tiempo y esfuerzo en realizarlo. Por eso, decidí aprovechar esta ocasión para empezar a conocer más acerca de este mundo legal que no me atrae demasiado pero que, por los motivos ya descritos, es interesante conocer. La decisión de qué licencia utilizar fue bastante sencilla de tomar. Buscaba empezar a adentrarme en el mundo de las licencias sin tener que pagar demasiado, ya que lo que realmente buscaba era

asegurar de una manera más formal mi autoría sobre el proyecto, sin importarme demasiado quién decidía finalmente hacer uso de él. Con esto, me incliné directamente a la Licencia Pública General, comúnmente referida por sus siglas en inglés GPL (*General Public Licence*) del proyecto GNU. Esta licencia es suficiente para atribuirme la autoría de este proyecto, tal y como yo quería. También resulta beneficioso para cualquier persona que buscase una aplicación con estas características que no quisiera gastar dinero; un principiante en Kotlin, programación Android o, en general, en el desarrollo de software que buscase un ejemplo práctico de una aplicación con estas características; y, finalmente, para una persona que pudiese estar interesado en comprobar cuál es mi metodología en el desarrollo de software. Todo esto es posible gracias al hecho de que la licencia GPL de GPU, además de concederme los derechos de autor del código, autoriza expresamente la explotación de la aplicación y la posibilidad de visualizar, modificar y distribuir el software con total libertad (siempre respetando la autoría, como se ha descrito anteriormente).

Pasando a hablar de la solución software propuesta para el problema descrito al inicio de este apartado, el aplicativo se tratará de una aplicación móvil para el sistema operativo Android desarrollada en el lenguaje de programación Kotlin que permitirá pasar lista a uno o más grupos de personas. Además de esta funcionalidad principal, la aplicación almacenará información adicional de las personas del grupo, como datos de contacto; y permitirá la exportación de los datos almacenados a un formato fácilmente visible en una aplicación de hoja de cálculo, permitiendo así al usuario elaborar un informe más detallado de asistencia si así lo requiriesen.

La metodología de desarrollo a seguir en este proyecto será un desarrollo incremental. El desarrollo incremental se caracteriza por priorizar la compleción un primer objetivo funcional, pasar a producción, e ir completando más adelante los objetivos funcionales restantes. Esto permite que, desde el momento que se despliega una primera versión de la aplicación, ésta ya cuenta con una parte funcional, permitiendo detectar errores o posibles desviaciones de los objetivos iniciales antes de que la toma de acciones pertinentes para subsanar estos errores sea demasiado costosa o complicada.

Esta decisión se tomó debido a los siguientes aspectos:

- **El proyecto no requiere coordinación.** Este proyecto es un desarrollo propio realizado por una única persona. Eso significa que la idea siempre es la misma y proviene de la misma persona, permitiendo así tirar hacia un tipo de desarrollo más tradicional frente a un desarrollo más en línea con los desarrollos ágiles y la toma de decisiones en equipo, como SCRUM.
- **Interés en tener una documentación formal.** En el desarrollo ágil también existe una documentación, pero es mucho más resumida e informal. Esto es porque parte de la filosofía ágil es la dar más importancia a la autodocumentación frente a la idea más tradicionalista de tener un documento como el presente que contenga toda la información formal del proyecto. Como, en nuestro caso, interesa destacar este documento, una vez más, interesa una metodología algo más tradicional.

- **Tradicional, pero flexible.** Aunque este tipo de desarrollo puede considerarse algo tradicional, ya se encuentra a medio camino entre la idea proveniente de la ingeniería industrial acerca de como afrontar un desarrollo software -también conocido como modelo en cascada- y las metodologías ágiles, que surgen de la necesidad de añadir un enorme grado de flexibilidad al desarrollo software, dadas sus características de cambio y evolución constante propias del mundo en el que vivimos.
- **La aplicación busca aportar utilidad.** Como ya se mencionó al inicio de esta sección, el objetivo de este proyecto es, desde sus orígenes, hacer frente a una necesidad real. Por este motivo, es interesante contar con una aplicación funcional, aunque dicha funcionalidad sea parcial, para ir solucionando este problema poco a poco, ya que sus características lo hacen posible e incluso conveniente.

Resumiendo, este proyecto y su documentación surgen a raíz de la necesidad de facilitar el control de asistencia en un grupo de alumnos de judo. A esta necesidad se le suma la posibilidad de hacer este proyecto público, para así potenciar mi currículum, mostrando mis habilidades y forma de trabajar. A mayores, añadimos la posibilidad de tener un primer contacto con las licencias y los derechos de autor. La metodología a utilizar en el proyecto será incremental debido a que es un desarrollo pequeño y realizado por una sola persona, existe un interés en hacer una documentación formal, alejándolo de metodologías más ágiles. También aporta una flexibilidad que no tiene el modelo tradicional o en cascada, que es interesante para absorber los peligros que supone la incertidumbre de eventos que pueden aparecer durante el proyecto. Finalmente, se busca contar con una versión funcional, aunque dicha funcionalidad sea parcial, en cuanto antes, para poder aprovechar el beneficio que la aplicación pretende aportar.

Todo esto dicho, pasamos a la siguiente sección, donde realizaremos un análisis de los requisitos del proyecto.

2. Análisis de requisitos

Al inicio de todo proyecto software, es importante conocer los requisitos del mismo, ya que condicionarán tanto la forma en la que se irá desarrollando, como el resultado final. En este apartado, describiremos los requisitos del sistema, así como sus objetivos funcionales. Para ser más concretos, en los siguientes subapartados veremos los objetivos funcionales, los requisitos funcionales, los requisitos no-funcionales y, finalmente, los requisitos de información.

Previo a empezar a describir los requisitos, explicaremos un poco la notación utilizada en este apartado. En primer lugar, es importante explicar que los requisitos suelen estar relacionados con los objetivos funcionales. Por este motivo, para facilitar en la identificación de los objetivos que son referenciados, los objetivos se referenciaran mediante etiquetas de la forma [OBJ-XX], donde "XX" es el número del objetivo. Los requisitos, por lo tanto, serán de la notación [R-XXYY], donde "XX" coincidirá con el objetivo funcional referenciado e "YY" será el número del requisito de ese mismo tipo asociado al objetivo.

Un ejemplo de esto sería: [RF-105], que sería el quinto requisito funcional (RF) del objetivo 1 ([OBJ-01]). Como se ha podido observar en el ejemplo, si el primer número del objetivo es "0", se omitirá en las etiquetas de los requisitos para mejorar la legibilidad.

2.1. Objetivos Funcionales

Este subapartado busca describir brevemente los objetivos funcionales del sistema. Los objetivos funcionales describen una serie de hitos o, como su nombre indica, objetivos generales que se espera conseguir con la aplicación. Esto nos permite definir una primera separación o modularización del proyecto, estableciendo una lista de requisitos segmentada, así aumentando su grado de legibilidad al añadir un mayor grado de organización.

Los objetivos funcionales son, habitualmente, un número reducido de apartados a partir de los cuales surgen los requisitos funcionales, requisitos no funcionales y requisitos de información, que aumentan la granularidad del objetivo funcional al añadirle un mayor grado de detalle.

Objetivos funcionales	
OBJ-01	Gestión de grupos e integrantes
OBJ-02	Realizar un control de asistencia de los integrantes de un grupo
OBJ-03	Exportar datos de un grupo

2.2. Requisitos funcionales

Este subapartado describe brevemente los requisitos funcionales de la aplicación. Los requisitos funcionales hacen referencia a una acción concreta que se espera que la aplicación pueda realizar. Como se describe en el glosario, en el análisis de requisitos suele emplearse la palabra sistema o el sistema para referirse a la aplicación, ya que en este punto nos encontramos en un nivel de abstracción independiente de cómo se va a llevar a cabo el proyecto, aunque ya hayamos decidido como hacerlo. Por este motivo, los requisitos funcionales suelen presentarse como una oración que tiene a "el sistema" como sujeto. Los requisitos funcionales describen las acciones o funciones que el sistema debe tener para cumplir los objetivos funcionales. En definitiva, describen el "qué hace el sistema".

A partir de los requisitos funcionales se desarrollarán en los apartados siguientes los casos de uso del sistema, ya que describen la funcionalidad concreta que debe ser proporcionada.

Requisitos Funcionales	
RF-101	El sistema permitirá crear un nuevo grupo
RF-102	El sistema permitirá añadir un nuevo integrante en un grupo
RF-103	El sistema permitirá modificar los datos de un integrante
RF-104	El sistema permitirá eliminar un integrante de un grupo
RF-105	El sistema permitirá eliminar un grupo
RF-106	El sistema permitirá visualizar los datos de un grupo
RF-107	El sistema permitirá visualizar los datos de un integrante
RF-201	El sistema permitirá realizar un control de asistencia sobre un grupo
RF-202	El sistema permitirá modificar un control de asistencia ya realizado
RF-301	El sistema permitirá exportar los datos de asistencia de un grupo

2.3. Requisitos no-funcionales

Este subapartado describe brevemente los requisitos no-funcionales de la aplicación. Los requisitos no funcionales establecen una serie de restricciones acerca de la forma en la cual se debe alcanzar el objetivo funcional con el que se asocian. Esto permite incluir como requisitos detalles que se escapan al ámbito de los requisitos funcionales, como pueden ser tecnologías, herramientas o metodologías concretas que se deben emplear en la realización de una acción necesaria para cumplir el objetivo, describiendo el "cómo lo hace el sistema".

Estos requisitos pueden modificar ligeramente la forma en la cual se realizan los casos de uso, pero su impacto es bastante pequeño en esta parte del proyecto. Su importancia se ve incrementada, sin embargo, en las últimas fases del análisis del proyecto, así como el desarrollo de la propia aplicación resultante.

Existen algunos requisitos funcionales, como el lenguaje de programación o un requisito proveniente del entorno en el que operará el sistema que afecta a la totalidad del sistema. Estos requisitos son importantes y deben quedar reflejados, a pesar de no relacionarse con ningún objetivo funcional. Estos requisitos no-funcionales en particular tendrán el número "0" como prefijo a su identificador.

Requisitos no-funcionales	
RNF-001	El sistema se desarrollará en el lenguaje de programación Kotlin
RNF-002	El sistema será compatible con la versión 8 del sistema operativo Android
RNF-101	Un grupo puede ser importado a la aplicación para crearlo
RNF-102	Un grupo podrá ser creado manualmente integrante a integrante
RNF-103	Un grupo puede ser importado en forma de texto que seguirá el formato indicado en el fichero *example.new_group.txt*
RNF-104	Un grupo puede ser importado desde un fichero CSV que seguirá el formato indicado en el fichero *example.new_group.csv* o *example.group.csv*
RNF-105	Un grupo puede ser importado desde un fichero XML que seguirá el formato indicado en el fichero *example.new_group.xml* o example.*group.xml*
RNF-106	Un fichero XML que represente un grupo seguirá la definición establecida en *crazyjaeger_rollcall_group.dtd*
RNF-107	Un grupo puede ser importado desde un fichero JSON que seguirá el formato indicado en el fichero *example.new_group.json* o *example.group.json*
RNF-103	La eliminación de un integrante no eliminará la información que se haya recogido del mismo
RNF-301	Los datos de un grupo se podrán exportar a un único fichero
RNF-302	Los datos de un grupo se podrán exportar a un fichero CSV que seguirá el formato indicado en el fichero *example.group.csv*
RNF-303	Los datos de un grupo se podrán exportar a un fichero XML que seguirá el formato indicado en el fichero *example.group.xml*
RNF-304	Un fichero XML que represente un grupo seguirá la definición establecida en *crazyjaeger_rollcall_group.dtd*
RNF-305	Los datos de un grupo se podrán exportar a un fichero JSON que seguirá el formato indicado en el fichero *example.group.json*

Los ficheros referentes a ejemplos referenciados en los requisitos no-funcionales podrán encontrarse en el directorio *example.data* que forma parte del proyecto y el archivo *crazyjaeger_rollcall_group.dtd* se encontrará presente en el directorio *doc* del proyecto. No obstante, para mayor comodidad, se incluirá en el anexo de este documento una copia del contenido de los ficheros referenciados.

A continuación, nos encontraríamos un apartado relativo a las llamadas Reglas de Negocio. Al igual que los requisitos no-funcionales, las reglas de negocio, provenientes del inglés *Buisness Rules*, establecen una serie de restricciones acerca de cómo debe realizar el sistema una acción concreta. La diferencia es que, al contrario de los

requisitos no-funcionales que establecen restricciones acerca de tecnologías o metodologías, las reglas de negocio establecen restricciones propias de la metodología utilizada en la organización que va a utilizar la herramienta. Si esta aplicación contara con diferentes roles de usuario, como una persona encargada de dar de alta a nuevos grupos y otra diferente encargada de llevar el control de asistencia, el requisito que marca la restricción de acciones realizables por cada rol iría en este apartado.

En el caso de esta aplicación, no necesitamos establecer una jerarquía de permisos y restricciones ni debemos ajustarnos a la forma de operar de una organización concreta. Debido a ello, no existirá esta sección de reglas de negocio, aunque si lo menciono para destacar que es bastante común encontrarlo en otros proyectos.

2.4. Requisitos de información

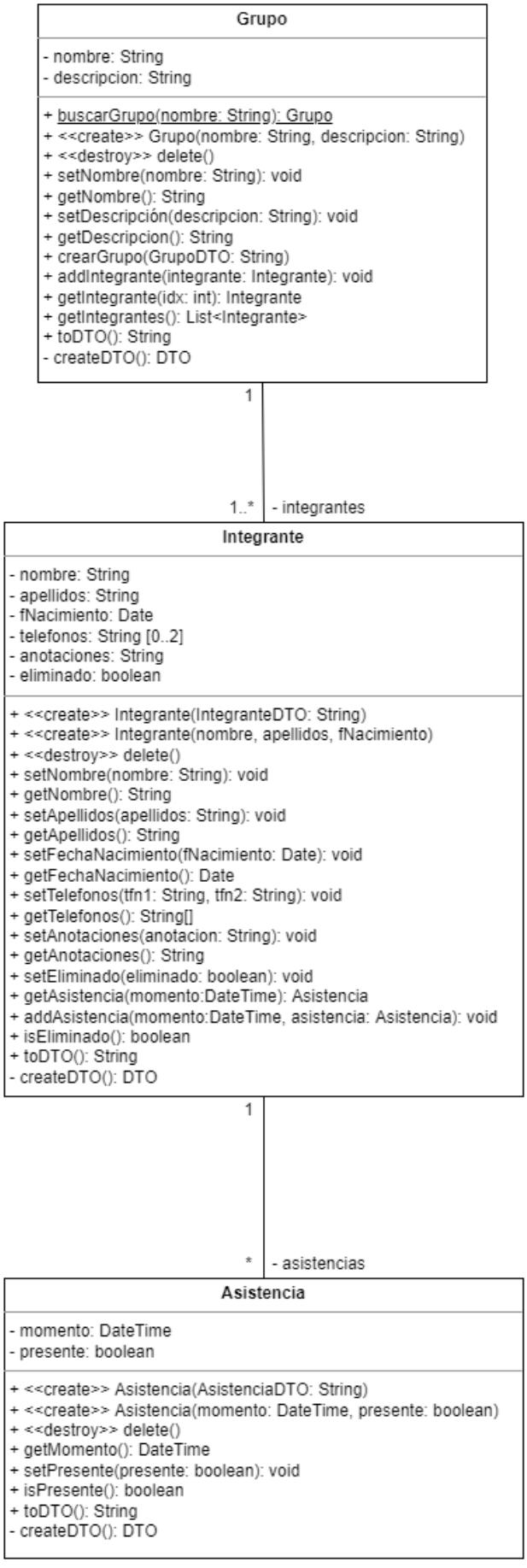
En este apartado describimos los requisitos de información. Los requisitos de información, en ocasiones también referidas como requisitos funcionales de información, establecen qué datos de los diferentes elementos presentes en el sistema se deben mantener durante el funcionamiento del mismo. Estos datos, adicionalmente, suelen coincidir con los datos que se deberán almacenar en memoria persistente posterior a su uso por el sistema. Mucho de lo que se incluye en esta serie de requisitos será lo que posteriormente se verá reflejado de una forma más gráfica en los diagramas de modelo de dominio, diagramas de entidad-relación o similares.

Los requisitos de información se suelen representar como un listado de la información que se debe manejar de cada uno de los elementos.

Requisitos de información	
RI-101	De cada grupo, el sistema almacenará: • El nombre del grupo • La descripción del grupo • Los integrantes que pertenecen al grupo
RI-102	De cada integrante, el sistema almacenará: • El nombre • Los apellidos • Fecha de nacimiento • Como máximo dos teléfonos de contacto • Anotaciones • Sus asistencias • Si ha sido eliminado o no del grupo
RI-201	De cada asistencia, el sistema almacenará: • El momento en el que se tomó la asistencia • Si el usuario estaba presente o no

3. Modelo de dominio

El modelo de dominio es un diagrama de clases que muestra de manera gráfica los elementos que forman parte del dominio del sistema. Estas clases surgen como resultado del análisis y podrán ser diferentes a las entidades reales que existirán en el sistema ya que el modelo de dominio permite abstraerse de las particularidades de la tecnología que se utilizará finalmente para desarrollar el sistema software. Además del diagrama, se incluirá un subapartado con las restricciones OCL (*Object Contraint Language*) aplicados al dominio.


Grupo
- nombre: String
- descripcion: String
+ buscarGrupo(nombre: String): Grupo
+ <<create>> Grupo(nombre: String, descripcion: String)
+ <<destroy>> delete()
+ setNombre(nombre: String): void
+ getNombre(): String
+ setDescripción(descripcion: String): void
+ getDescripcion(): String
+ crearGrupo(GrupoDTO: String)
+ addIntegrante(integrante: Integrante): void
+ getIntegrante(idx: int): Integrante
+ getIntegrantes(): List<Integrante>
+ toDTO(): String
- createDTO(): DTO
1
1..*
- integrantes
Integrante
- nombre: String
- apellidos: String
- fNacimiento: Date
- telefonos: String [0..2]
- anotaciones: String
- eliminado: boolean
+ <<create>> Integrante(IntegranteDTO: String)
+ <<create>> Integrante(nombre, apellidos, fNacimiento)
+ <<destroy>> delete()
+ setNombre(nombre: String): void
+ getNombre(): String
+ setApellidos(apellidos: String): void
+ getApellidos(): String
+ setFechaNacimiento(fNacimiento: Date): void
+ getFechaNacimiento(): Date
+ setTelefonos(tfn1: String, tfn2: String): void
+ getTelefonos(): String[]
+ setAnotaciones(anotacion: String): void
+ getAnotaciones(): String
+ setEliminado(eliminado: boolean): void
+ getAsistencia(momento:DateTime): Asistencia
+ addAsistencia(momento:DateTime, asistencia: Asistencia): void
+ isEliminado(): boolean
+ toDTO(): String
- createDTO(): DTO
1
*
- asistencias
Asistencia
- momento: DateTime
- presente: boolean
+ <<create>> Asistencia(AsistenciaDTO: String)
+ <<create>> Asistencia(momento: DateTime, presente: boolean)
+ <<destroy>> delete()
+ getMomento(): DateTime
+ setPresente(presente: boolean): void
+ isPresente(): boolean
+ toDTO(): String
- createDTO(): DTO

3.1. Restricciones OCL

Las restricciones OCL establecen una serie de restricciones sobre los datos del dominio. Estas restricciones pueden ser invariantes, o pueden representar precondiciones y postcondiciones de la ejecución de una operación determinada.

```
Context: Grupo
inv:
nombre <> null and self.getNombre() = self.nombre and
self.getDescripcion() = self.descripcion and
self.getIntegrantes() = integrantes
```

```
Context: Grupo::Grupo(nombre, descripción)
Pre:
Grupo.buscarGrupo(nombre) = NULL
Post:
Grupo.buscarGrupo(nombre) <> NULL and
Grupo.buscarGrupo(nombre) -> result.getNombre() = nombre and
result.getDescripcion() = descripcion
```

```
Context: Grupo::delete()
Post:
Grupo.buscarGrupo(nombre) = NULL
```

```
Context: Grupo::setNombre(nombre)
Inv:
nombre <> NULL
Post:
self.nombre = nombre
```

```
Context: Grupo::setDescripcion(descripcion)
Post:
Self.descripcion = descripción
```

```
Context: Grupo::crearGrupo(grupoDTO)
Inv:
grupoDTO <> NULL
```

```
Context: Grupo::addIntegrante(integrante)
Inv:
integrante <> NULL
Post:
self.integrantes -> exists(integrante)
```

```
Context: Grupo::getIntegrante(idx)
Inv:
idx >= 0 and idx < self.integrantes -> size()
post:
result = self.integrantes -> get(idx)
```

```
Context: Integrante
Inv:
nombre <> NULL and apellidos <> NULL and fNacimiento <> NULL and
self.getNombre() = self.nombre and
self.getApellidos() = self.apellidos and
self.getFechaNacimiento() = self.fNacimiento and
self.getTelefonos() = self.telefonos and
self.getAnotaciones() = self.anotaciones and
self.isEliminado() = self.eliminado
```

```
Context: Integrante::Integrante(nombre, apellidos, fNacimiento)
Inv:
self. Integrante(nombre, apellidos, fNacimiento) implies
grupo.addIntegrante(self)
```

```
Context: Integrante::delete()
Post:
Grupo.integrantes -> NOT exists(self)
```

```
Context: Integrante::setNombre(nombre)
Inv:
nombre <> NULL
post:
self.nombre = nombre
```

```
Context: Integrante::setApellidos(apellidos)
Inv:
apellidos <> NULL
post:
self.apellidos = apellidos
```

```
Context: Integrante::setFechaNacimiento(fNacimiento)
Inv:
fNacimiento <> NULL
post:
self.fNacimiento = fNacimiento
```

```
Context: Integrante::setTelefonos(tfn1, tfn2)
Post:
self.telefonos = tfn1 + tfn2
```

```
Context: Integrante::setAnotaciones(anotacion)
Post:
self.anotaciones = anotacion
```

```
Context: Integrante::setEliminado(eliminado)
Post:
self.eliminado = eliminado
```

```
Context: Integrante::getAsistencia(momento)
Inv:
momento <> NULL and
asistencias -> exists(asistencia.getMomento() = momento) implies
result = asistencia and
asistencias -> NOT exists(asistencia.getMomento() = momento) implies
result = NULL
```

```
Context: Integrante::addAsistencia(momento, asistencia)
Inv:
momento <> NULL and asistencia <> NULL
post:
self.getAsistencia(momento) = asistencia
```

```
Context: Asistencia
Inv:
momento <> NULL and self.getMomento() = self.momento and
self.isPresente() = self.presente
```

```
Context: Asistencia::delete()
Post:
Self.integrante.getAsistencia(self.momento) = NULL
```

```
Context: Asistencia.setPresente(presente)
Post:
self.presente = presente
```

4. Persistencia

Un tema importante a tratar es el del almacenamiento de datos persistentes. Dada la naturaleza del sistema que pretendemos desarrollar con este proyecto, está claro que los datos recogidos tendrán que guardarse en memoria persistente para su uso tanto para desarrollar la función principal de la aplicación, que es pasar lista a un grupo de personas, como para permitir su exportación futura para fines estadísticos.

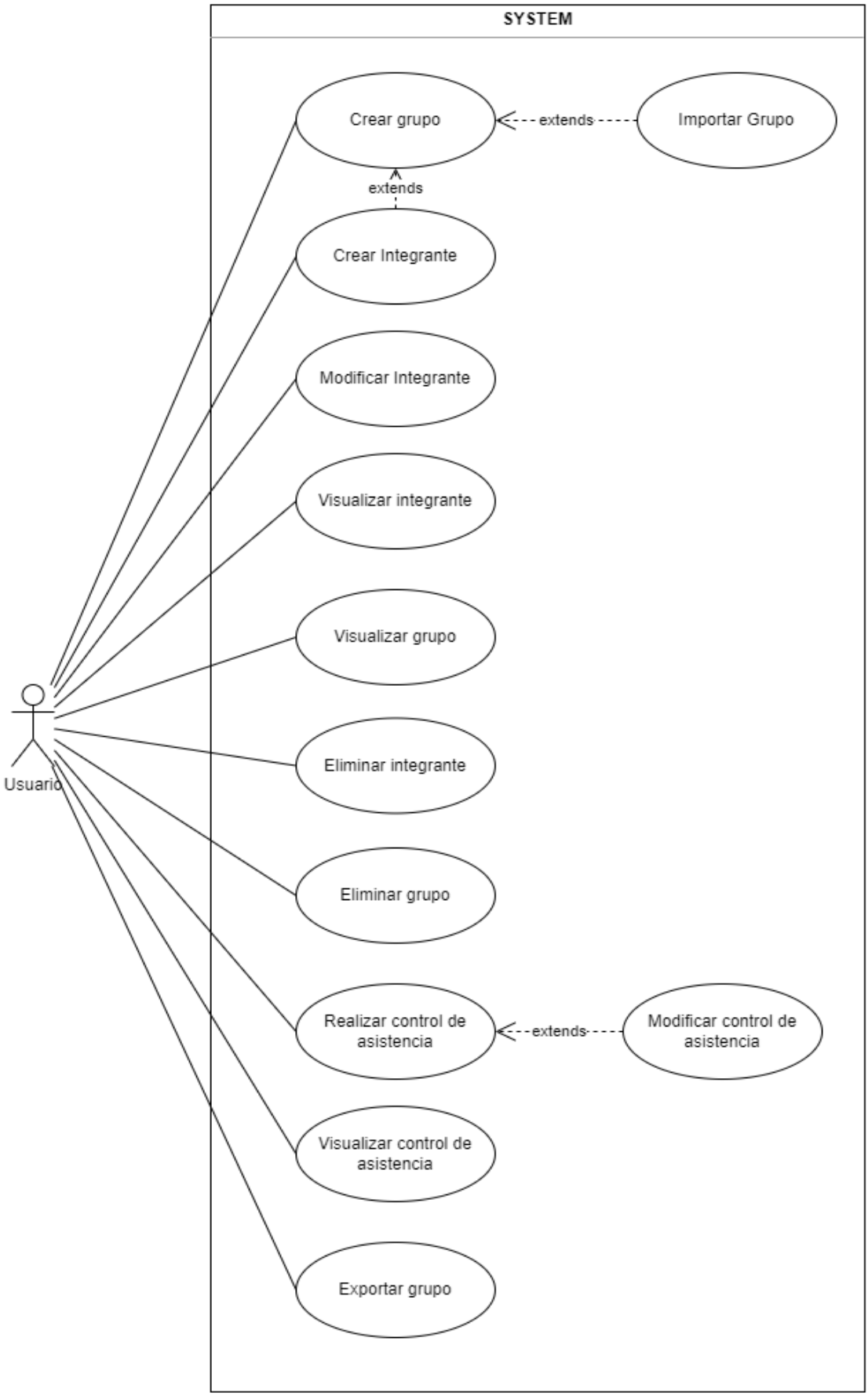
Aunque en un inicio se consideró la utilización de una base de datos SQLite para el almacenamiento persistente de los datos de la aplicación, se decidió optar por una alternativa debido a la cantidad de gente en línea que lo desaconseja, así como la propia experiencia personal con otras aplicaciones que hacen uso de ella, las cuales resultaban ser algo engorrosas y difíciles de entender en cuanto a la carga y descarga de datos desde la base de datos.

Para toda la aplicación, existirá un único fichero llamado *group_index.json* que actuará como un índice. Para cada grupo, almacenando tanto el nombre simple del grupo (el que se mostrará al usuario) como el nombre del fichero que guarda los datos relativos a ese grupo. Ese fichero tendrá un nombre parecido a *grp_group.json* para que sea fácil de distinguir en el sistema de archivos. Puede encontrarse un ejemplo de cada uno de estos dos ficheros en el directorio de ejemplos del proyecto, siendo éstos *example.group_index.json* y *example.grp_group.json*. También se adjuntará una copia del contenido en el anexo de este documento.

5. Casos de uso

Los casos de uso detallan las operaciones que pueden realizarse con la aplicación y surgen como resultado de los requisitos funcionales del sistema. Este apartado cuenta a su vez con tres subsecciones. El primero incluye un diagrama de casos de uso, que muestra gráficamente los casos de uso y los actores involucrados. La segunda subsección detalla el desarrollo de cada caso de uso. El tercer y último apartado muestra mediante diagramas de secuencia las operaciones realizadas según los casos de uso detallados.

5.1. Diagrama de casos de uso



5.2. Casos de uso detallados

5.2.1. Caso de uso: Crear Grupo

Caso de uso 1: Crear Grupo	
1.	Usuario solicita crear un grupo
2.	Sistema solicita al usuario que indique nombre y descripción del grupo
3.	Usuario indica nombre y descripción del grupo
4.	Sistema solicita al usuario que indique si desea crear el grupo manualmente o si desea importar el grupo
5.	Usuario indica que desea crear el grupo manualmente
6.	Sistema solicita que el usuario cree un nuevo integrante
7.	Comienza el *Caso de uso 3: Crear Integrante*
8.	Sistema solicita al usuario que indique si ha terminado de crear integrantes
9.	Usuario indica que ha terminado de crear integrantes
10.	Sistema muestra al usuario un resumen del grupo
11.	Sistema solicita al usuario que confirme la creación del grupo
12.	Usuario confirma la creación del grupo
13.	Sistema indica al usuario que el grupo se ha creado con éxito
Finaliza el caso de uso	

Excepción A: Usuario cancela la operación	
Ocurrencia en el paso 3, 5, 7, 9 y 12	
A.1.	Sistema indica al usuario que la operación se ha cancelado con éxito
Finaliza el caso de uso sin efecto	

Excepción B: Usuario indica que desea importar un grupo	
Ocurrencia en el paso 5	
B.1.	Comienza el *Caso de uso 2: Importar Grupo*
Continúa el caso de uso en el paso 10	

Excepción C: Usuario cancela la ejecución del *Caso de uso 3: Crear Integrante*	
Ocurrencia en el paso 7	
C.1.	Sistema comprueba que hay al menos un integrante en el grupo
Continúa el caso de uso en el paso 8	

Excepción D: Sistema comprueba que no hay ningún integrante en el grupo
Ocurrencia en el paso C.1
Continúa el caso de uso en el paso 4

Excepción E: Usuario cancela la ejecución del *Caso de uso 2: Importar Grupo*
Ocurrencia en el paso B.1
Continúa el caso de uso en el paso 4

Excepción F: Usuario indica que no ha terminado de crear integrantes
Ocurrencia en el paso 9
Continúa el caso de uso en el paso 6

5.2.2. Caso de uso: Importar Grupo

Caso de uso 2: Importar Grupo	
1.	Usuario solicita importar un grupo
2.	Sistema solicita al usuario que indique el lugar del que desea importar el grupo
3.	Usuario indica el lugar del que desea importar el grupo
4.	Sistema comprueba que el lugar indicado contiene un grupo importable
5.	Sistema importa el grupo
6.	Sistema indica al usuario que el grupo ha sido importado con éxito
Finaliza el caso de uso	

Excepción A: Usuario cancela la operación	
Ocurrencia en el paso 3	
A.1.	Sistema indica al usuario que la operación se ha cancelado con éxito
Finaliza el caso de uso sin efecto	

Excepción B: El lugar indicado por el usuario no contiene un grupo importable	
Ocurrencia en el paso 4	
B.1.	Sistema indica al usuario que no ha sido posible importar el grupo
B.2.	Sistema indica al usuario el motivo por el cual el grupo no se ha podido importar
Finaliza el caso de uso sin efecto	

5.2.3. Caso de uso: Crear Integrante

Caso de uso 3: Crear Integrante	
Precondición: el grupo al cual se va a añadir el integrante ha de estar seleccionado	
1.	Usuario solicita crear un nuevo integrante
2.	Sistema solicita al usuario que indique el nombre, apellidos y fecha de nacimiento del integrante
3.	Usuario indica el nombre, apellidos y fecha de nacimiento del integrante
4.	Sistema solicita al usuario que indique los números de teléfono del integrante
5.	Usuario indica los números de teléfono del integrante
6.	Sistema solicita al usuario que incluya una anotación
7.	Usuario incluye una anotación
8.	Sistema muestra al usuario un resumen del integrante
9.	Sistema solicita al usuario que indique si está correcto
10.	Usuario indica que los datos son correctos
11.	Sistema crea el integrante y lo añade al grupo seleccionado
Finaliza el caso de uso	

Excepción A: Usuario cancela la operación	
Ocurrencia en el paso 3, 5, 7 y 10	
A.1.	Sistema indica al usuario que la operación se ha cancelado con éxito
Finaliza el caso de uso sin efecto	

Excepción B: Usuario indica que los datos no son correctos
Ocurrencia en el paso 10
Continúa caso de uso en el paso 2

5.2.4. Caso de uso: Modificar Integrante

Caso de uso 4: Modificar Integrante	
Precondición: el grupo al cual pertenece el integrante ha de estar seleccionado	
1.	Usuario solicita modificar un integrante
2.	Sistema muestra al usuario los datos actuales del integrante
3.	Sistema solicita al usuario que indique el dato que desea modificar
4.	Usuario indica el dato que desea modificar
5.	Sistema solicita al usuario que indique el nuevo dato
6.	Usuario indica el nuevo dato
7.	Sistema muestra al usuario los datos tras la modificación
8.	Sistema solicita al usuario que indique si ha finalizado
9.	Usuario indica que ha finalizado
10.	Sistema guarda la modificación
Finaliza el caso de uso	

Excepción A: Usuario cancela la operación	
Ocurrencia en el paso 4,6 y 8	
A.1.	Sistema indica al usuario que la operación se ha cancelado con éxito
Finaliza el caso de uso sin efecto	

Excepción B: Usuario indica que no ha finalizado
Ocurrencia en el paso 9
Continúa caso de uso en el paso 3

5.2.5. Caso de uso: Visualizar Grupo

Caso de uso 5: Visualizar Grupo	
1.	Usuario solicita visualizar un grupo
2.	Sistema solicita al usuario que indique qué grupo desea visualizar
3.	Usuario indica el grupo que desea visualizar
4.	Sistema muestra los datos del grupo seleccionado
Finaliza el caso de uso	

Excepción A: Usuario cancela la operación	
Ocurrencia en el paso 3	
A.1.	Sistema indica al usuario que la operación se ha cancelado con éxito
Finaliza el caso de uso sin efecto	

5.2.6. Caso de uso: Visualizar integrante

Caso de uso 6: Visualizar Integrante	
Precondición: el grupo del cual se va a mostrar el integrante ha de estar seleccionado	
1.	Usuario solicita visualizar un integrante
2.	Sistema solicita al usuario que indique qué integrante del grupo desea visualizar
3.	Usuario indica el integrante del grupo que desea visualizar
4.	Sistema muestra los datos del integrante
Finaliza el caso de uso	

Excepción A: Usuario cancela la operación	
Ocurrencia en el paso 3	
A.1.	Sistema indica al usuario que la operación se ha cancelado con éxito
Finaliza el caso de uso sin efecto	

5.2.7. Caso de uso: Eliminar Integrante

Caso de uso 7: Eliminar Integrante	
Precondición: el grupo del cual se va a eliminar el integrante ha de estar seleccionado	
1.	Usuario solicita eliminar un integrante
2.	Sistema solicita al usuario que indique el integrante del grupo que desea eliminar
3.	Usuario indica el integrante del grupo que desea eliminar
4.	Sistema muestra al usuario los datos del integrante seleccionado
5.	Sistema solicita al usuario que confirme la eliminación del integrante
6.	Usuario confirma la eliminación del integrante
7.	Sistema elimina al usuario de la lista de integrantes actuales del grupo
Finaliza el caso de uso	

Excepción A: Usuario cancela la operación	
Ocurrencia en el paso 3 y 6	
A.1.	Sistema indica al usuario que la operación se ha cancelado con éxito
Finaliza el caso de uso sin efecto	

5.2.8. Caso de uso: Eliminar Grupo

Caso de uso 8: Eliminar Grupo	
1.	Usuario solicita eliminar un grupo
2.	Sistema solicita al usuario que indique el grupo que desea eliminar
3.	Usuario indica el grupo que desea eliminar
4.	Sistema muestra al usuario los datos del grupo seleccionado
5.	Sistema solicita al usuario que confirme la eliminación del grupo
6.	Usuario confirma la eliminación del grupo
7.	Sistema elimina el grupo
Finaliza el caso de uso	

Excepción A: Usuario cancela la operación	
Ocurrencia en el paso 3 y 6	
A.1.	Sistema indica al usuario que la operación se ha cancelado con éxito
Finaliza el caso de uso sin efecto	

5.2.9. Caso de uso: Realizar control de asistencia

Caso de uso 9: Realizar control de asistencia	
1.	Usuario solicita realizar un control de asistencia
2.	Sistema solicita al usuario que indique el grupo sobre el cual va a realizar el control de asistencia
3.	Usuario indica el grupo sobre el cual va a realizar un control de asistencia
4.	Sistema solicita al usuario que indique la fecha y hora del control
5.	Usuario indica la fecha y hora del control
6.	Sistema comprueba que el grupo seleccionado no tiene un control realizado en esa fecha y esa hora
7.	Sistema muestra el nombre y apellidos del siguiente integrante del grupo
8.	Sistema solicita al usuario que indique si el integrante está presente o no
9.	Usuario indica si el integrante está presente o no
10.	Sistema comprueba que el grupo no tiene más integrantes
11.	Sistema muestra al usuario el resumen del control de asistencia
12.	Sistema solicita al usuario que confirme el control de asistencia
13.	Usuario confirma el control de asistencia
14.	Sistema guarda el control de asistencia
Finaliza el caso de uso	

Excepción A: Usuario cancela la operación	
Ocurrencia en el paso 3, 5, 9 y 13	
A.1.	Sistema indica al usuario que la operación se ha cancelado con éxito
Finaliza el caso de uso sin efecto	

Excepción B: Si el grupo ya tiene un control para la fecha y hora indicados	
Ocurrencia en el paso 6	
B.1.	Sistema indica al usuario que ya se ha realizado un control de asistencia para en la fecha y horas seleccionados
Finaliza el caso de uso. Comienza C*aso de uso 10: Modificar control de asistencia*	

Excepción C: Sistema comprueba que el grupo tiene más integrantes
Ocurrencia en el paso 10
Continúa caso de uso en el paso 7

5.2.10. Caso de uso: Modificar control de asistencia

Caso de uso 10: Modificar control de asistencia	
Precondición: ya existe un control de asistencia para la fecha y hora indicados por el usuario en el *Caso de uso 9: Realizar control de asistencia*	
1.	Usuario solicita modificar un control de asistencia
2.	Sistema muestra al usuario el resumen del control de asistencia
3.	Sistema solicita al usuario que indique el integrante para el cual desea modificar la asistencia
4.	Usuario indica el integrante para el cual desea modificar la asistencia
5.	Sistema muestra al usuario el resumen del control de asistencia tras la modificación
6.	Sistema solicita al usuario que confirme la modificación
7.	Usuario confirma la modificación
8.	Sistema guarda la modificación
Finaliza el caso de uso	

Excepción A: Usuario cancela la operación	
Ocurrencia en el paso 4 y 7	
A.1.	Sistema indica al usuario que la operación se ha cancelado con éxito
Finaliza el caso de uso sin efecto	

Excepción B: El usuario no confirma la modificación
Ocurrencia en el paso 7
Continúa caso de uso en el paso 3

5.2.11. Caso de uso: Visualizar control de asistencia

Caso de uso 10: Visualizar control de asistencia	
Precondición: el grupo para el cual se va a visualizar el control de asistencia se encuentra seleccionado	
1.	Usuario solicita visualizar un control de asistencia
2.	Sistema solicita al usuario que indique el control de asistencia que desea visualizar
3.	Usuario indica el control de asistencia que desea visualizar
4.	Sistema muestra al usuario los datos del control de asistencia
Finaliza el caso de uso	

Excepción A: Usuario cancela la operación	
Ocurrencia en el paso 3	
A.1.	Sistema indica al usuario que la operación se ha cancelado con éxito
Finaliza el caso de uso sin efecto	

5.2.12. Caso de uso: Exportar Grupo

Caso de uso 12: Exportar Grupo	
1.	Usuario solicita exportar un grupo
2.	Sistema solicita al usuario que indique que grupo desea exportar
3.	Usuario indica el grupo que desea exportar
4.	Sistema solicita al usuario que indique el lenguaje de persistencia que desea emplear para la exportación
5.	Usuario indica el lenguaje de persistencia que desea emplear para la exportación
6.	Sistema solicita al usuario que indique el lugar en el cual desea que se guarde la exportación
7.	Usuario indica el lugar en el cual desea que se guarde la exportación
8.	Sistema exporta los datos del grupo en el lugar indicado utilizando el lenguaje de persistencia indicado
Finaliza el caso de uso	

Excepción A: Usuario cancela la operación	
Ocurrencia en el paso 3, 5 y 7	
A.1.	Sistema indica al usuario que la operación se ha cancelado con éxito
Finaliza el caso de uso sin efecto	

5.2.13. Caso de uso: Modificar Grupo

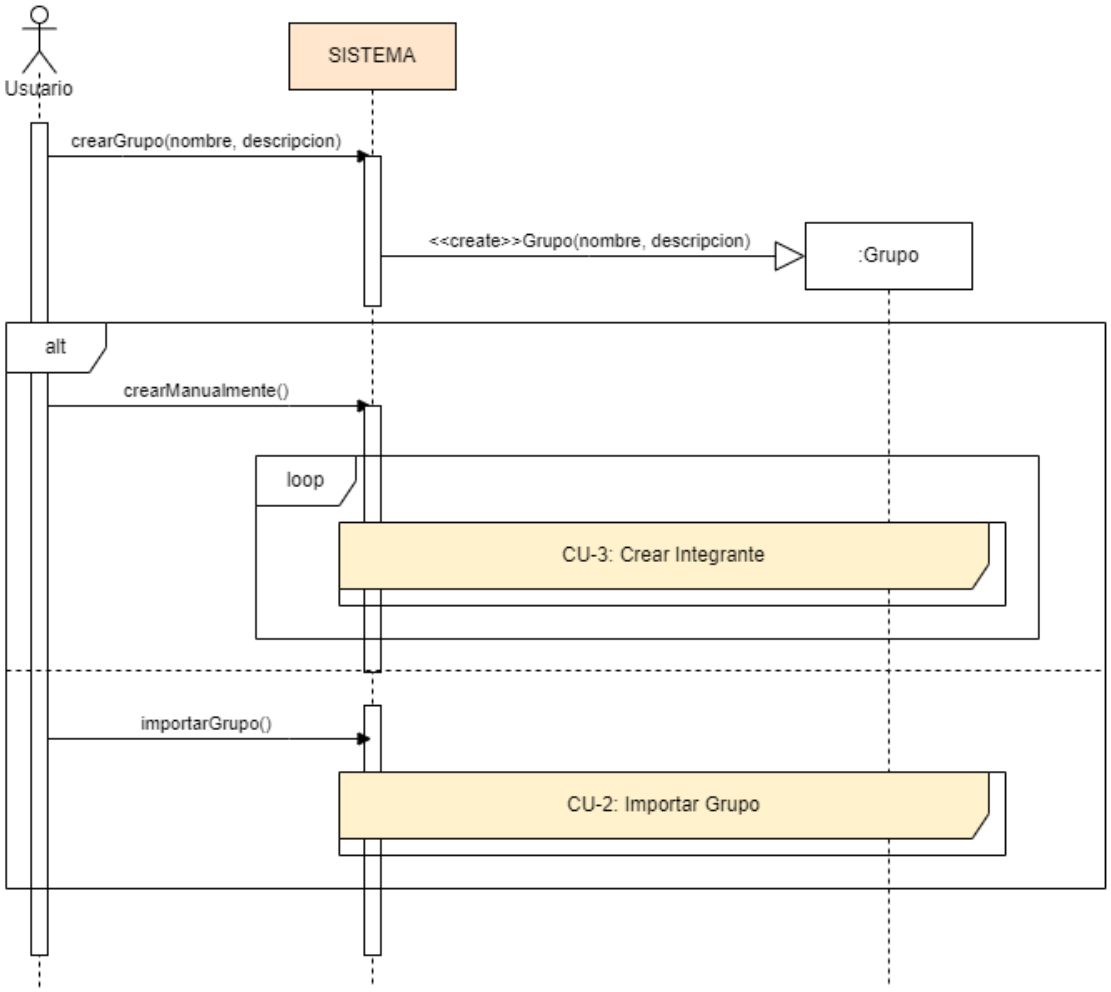
Caso de uso 13: Modificar Grupo	
1.	Usuario solicita modificar un grupo
2.	Sistema muestra al usuario los datos actuales del grupo
3.	Sistema solicita al usuario que indique el dato que desea modificar
4.	Usuario indica el dato que desea modificar
5.	Sistema solicita al usuario que indique el nuevo dato
6.	Usuario indica el nuevo dato
7.	Sistema muestra al usuario los datos tras la modificación
8.	Sistema solicita al usuario que indique si ha finalizado
9.	Usuario indica que ha finalizado
10.	Sistema guarda la modificación
Finaliza el caso de uso	

Excepción A: Usuario cancela la operación	
Ocurrencia en el paso 4, 6 y 8	
A.1.	Sistema indica al usuario que la operación se ha cancelado con éxito
Finaliza el caso de uso sin efecto	

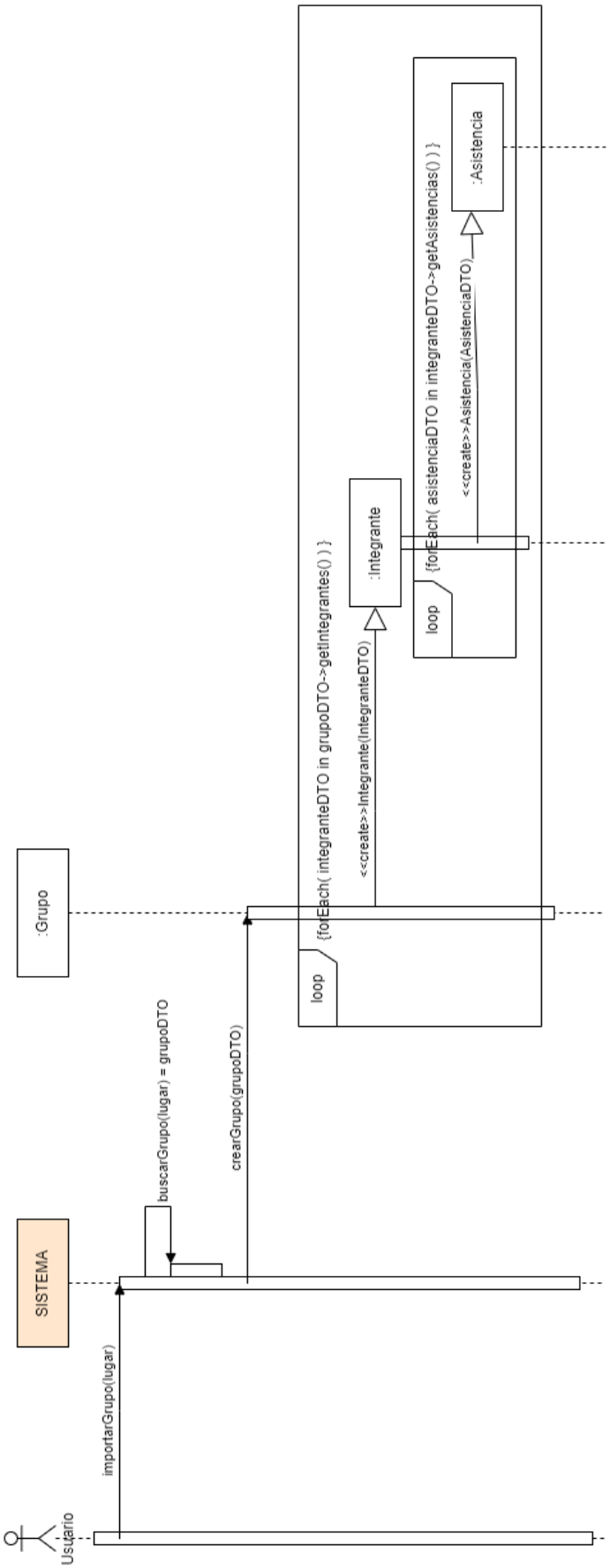
Excepción B: Usuario indica que no ha finalizado
Ocurrencia en el paso 9
Continúa caso de uso en el paso 3

5.3. Diagramas de secuencia

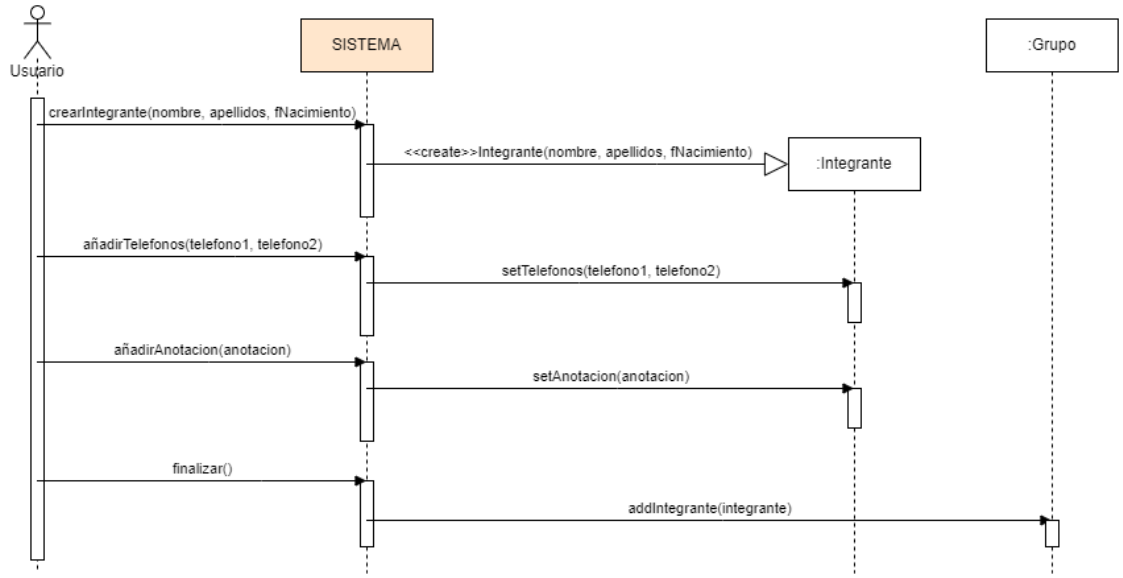
5.3.1. Caso de uso: Crear Grupo



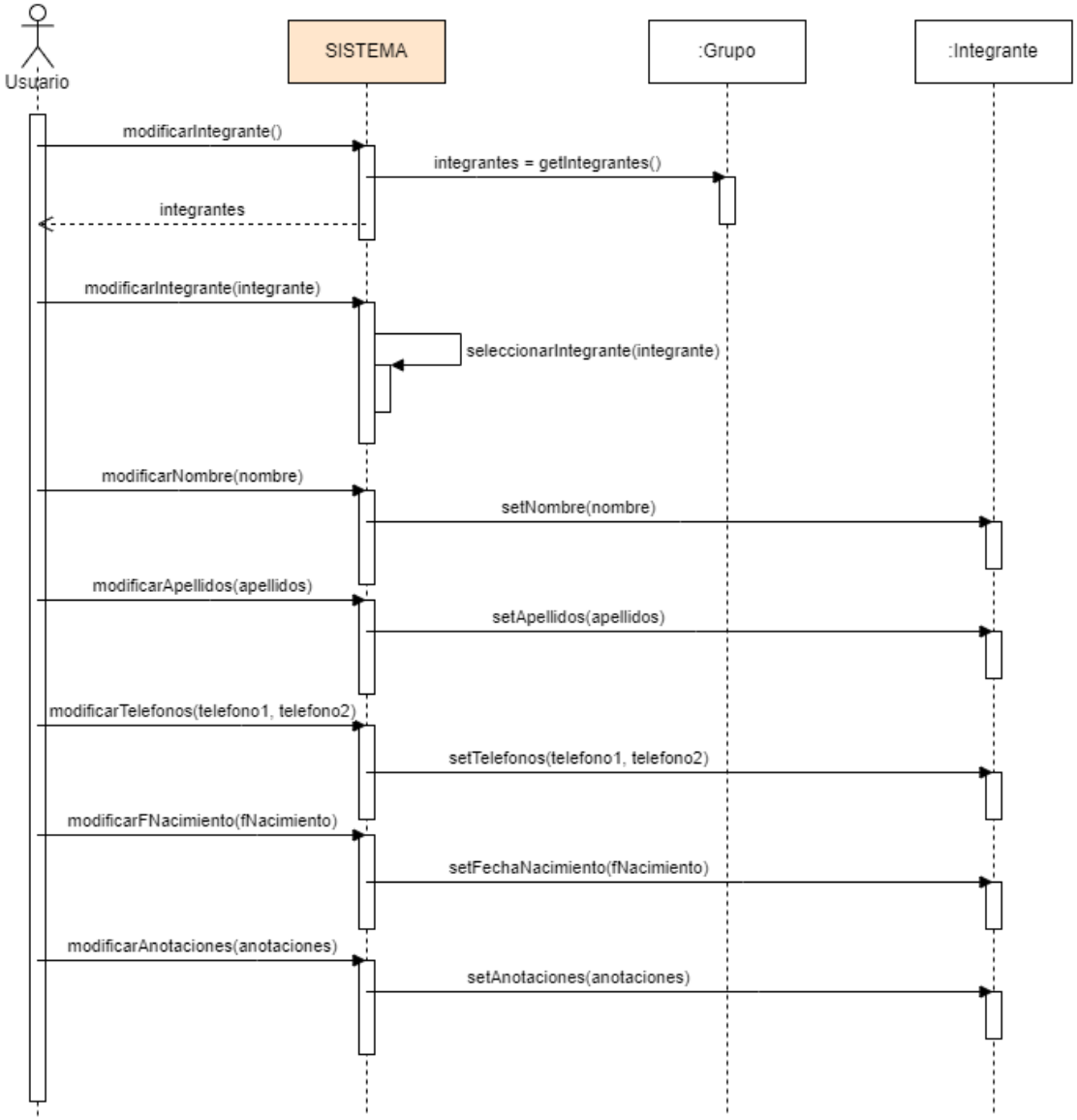
5.3.2. Caso de uso: Importar Grupo



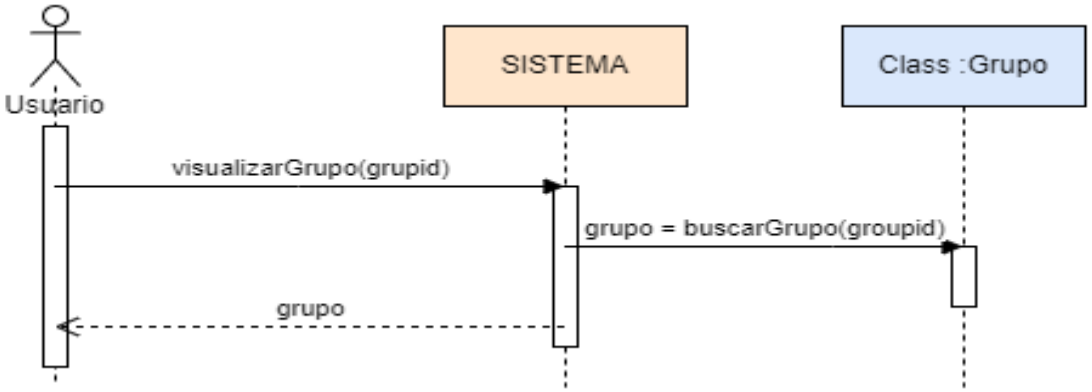
5.3.3. Caso de uso: Crear Integrante



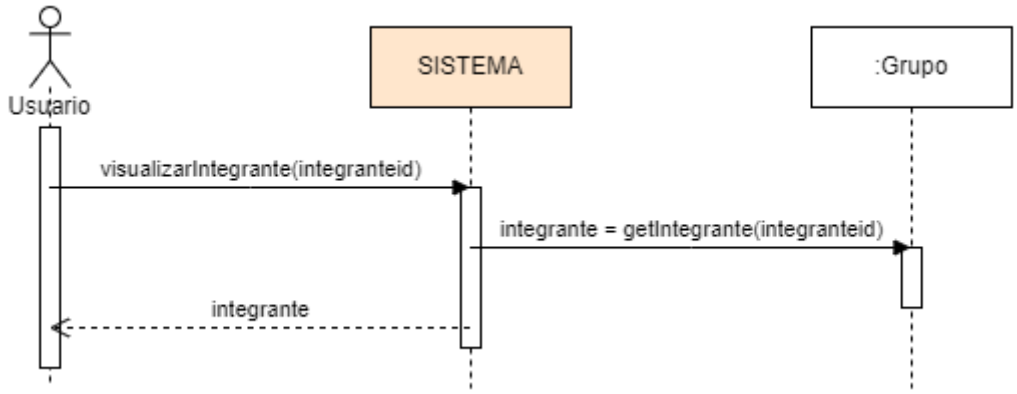
5.3.4. Caso de uso: Modificar Integrante



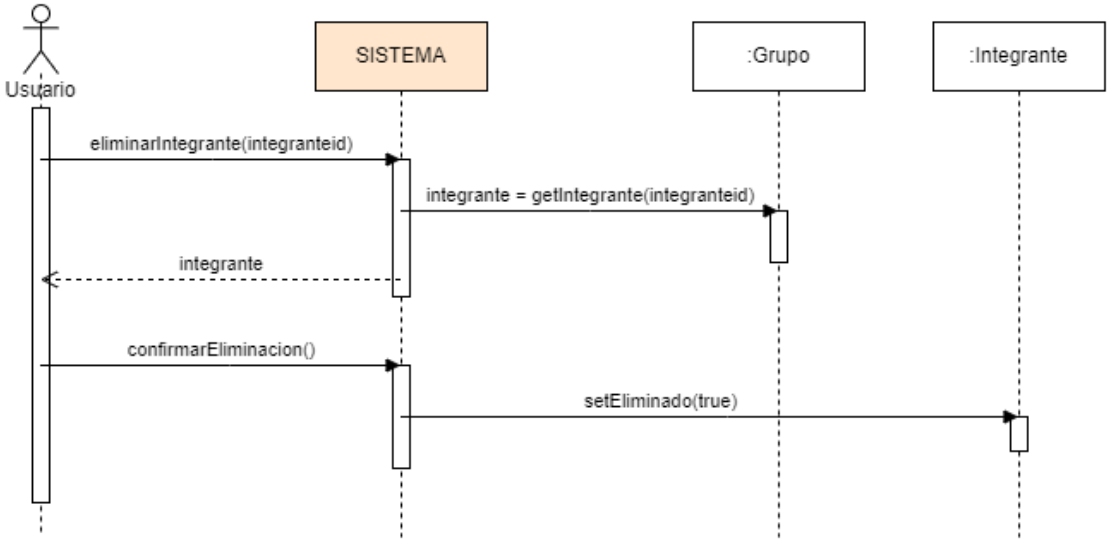
5.3.5. Caso de uso: Visualizar Grupo



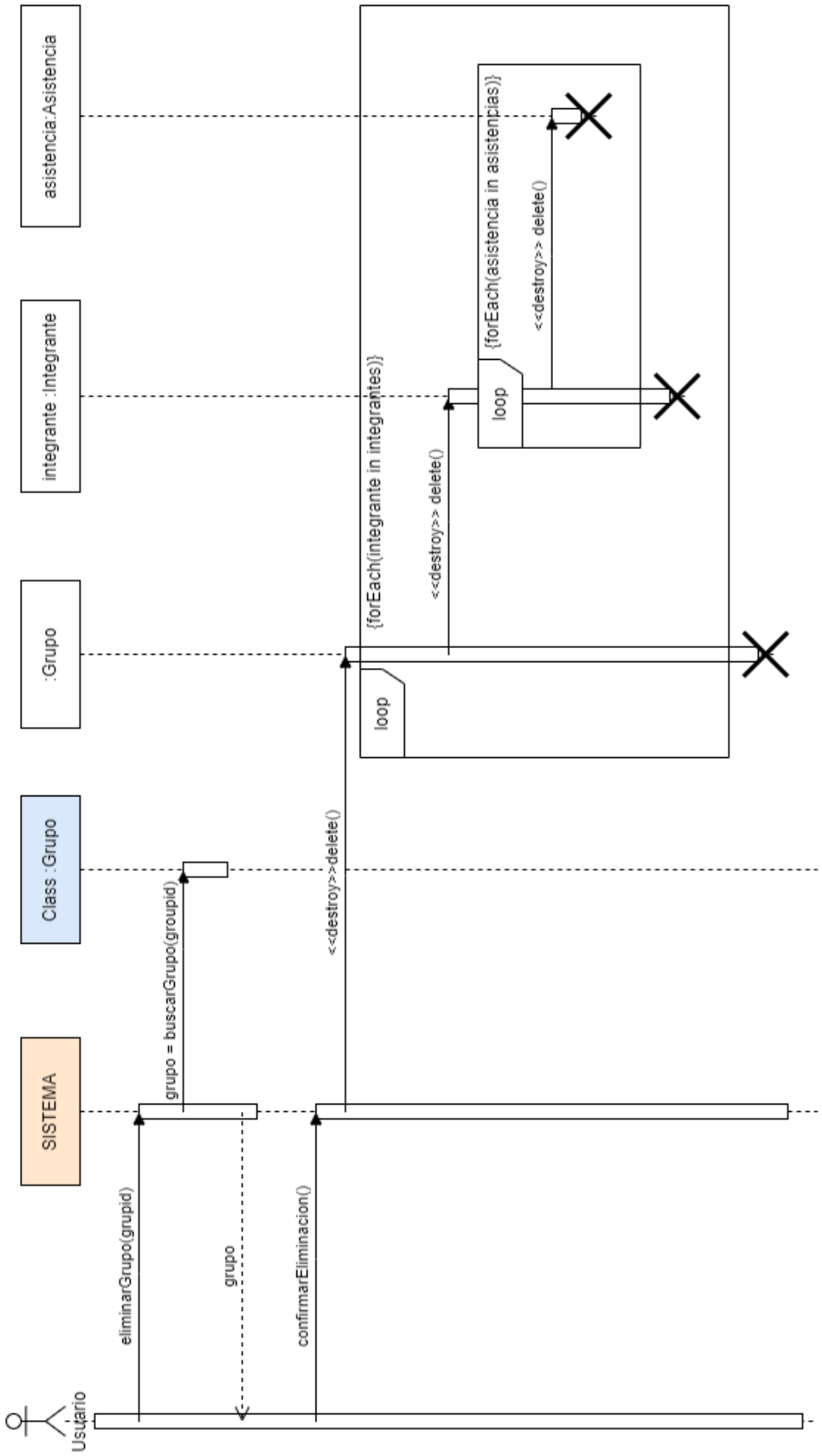
5.3.6. Caso de uso: Visualizar Integrante



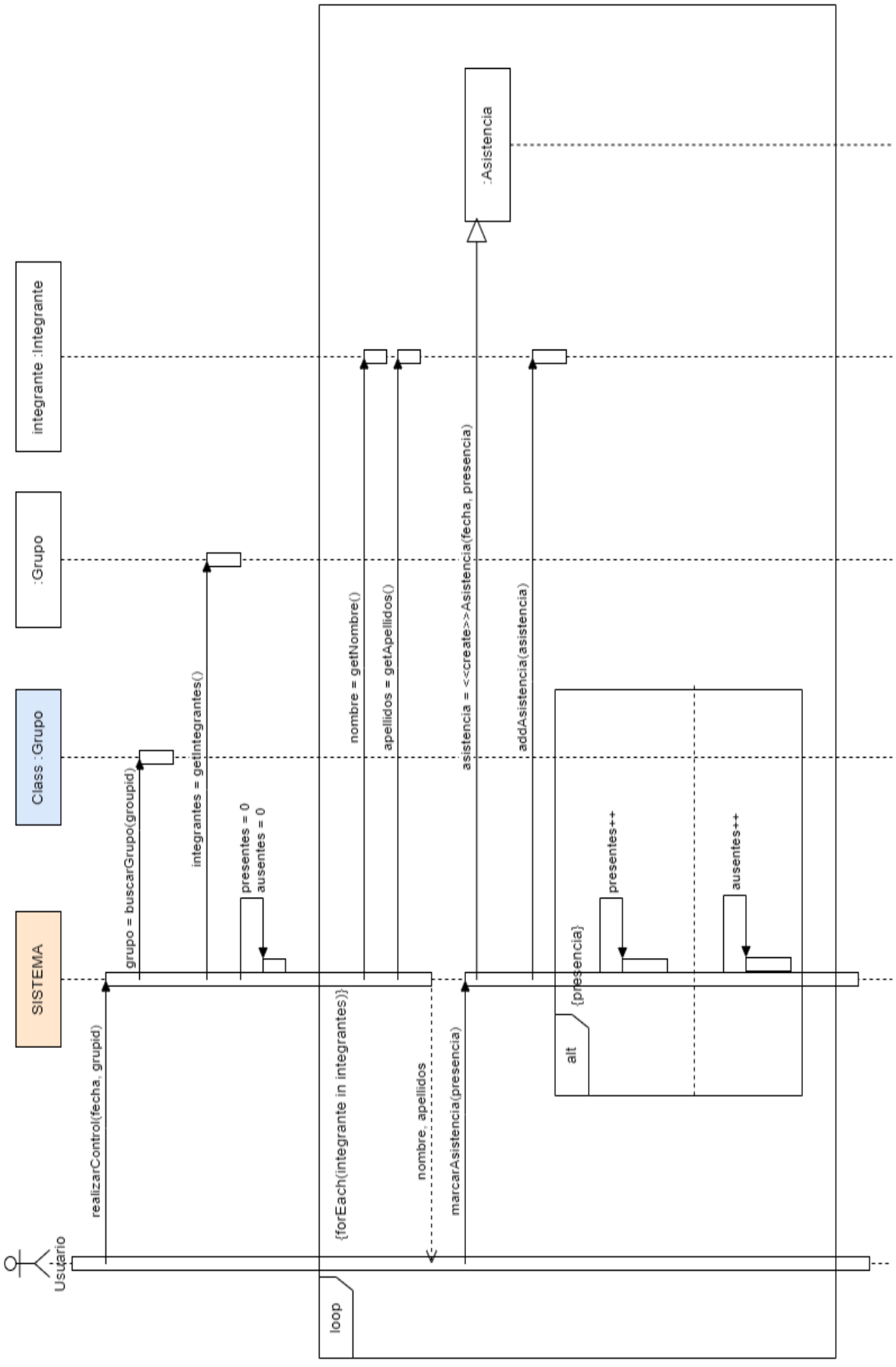
5.3.7. Caso de uso: Eliminar Integrante



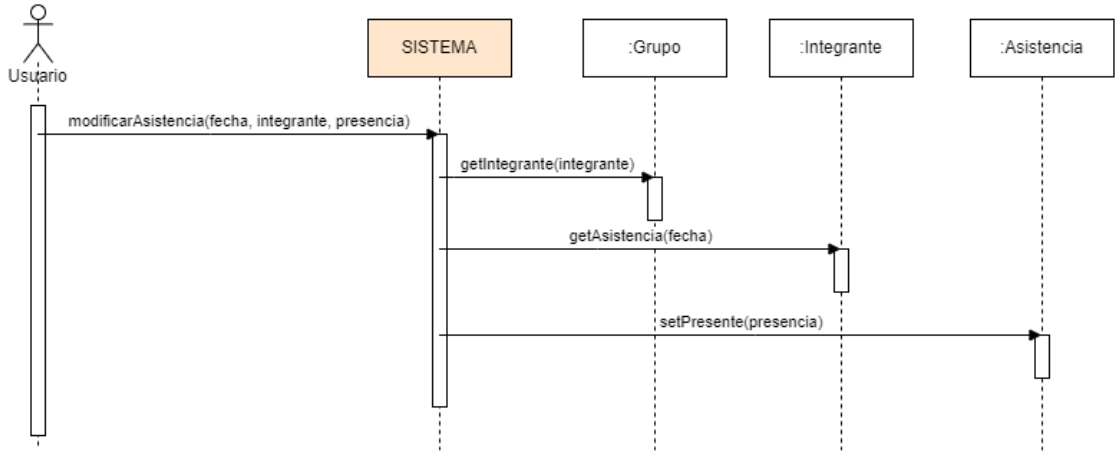
5.3.8. Caso de uso: Eliminar Grupo



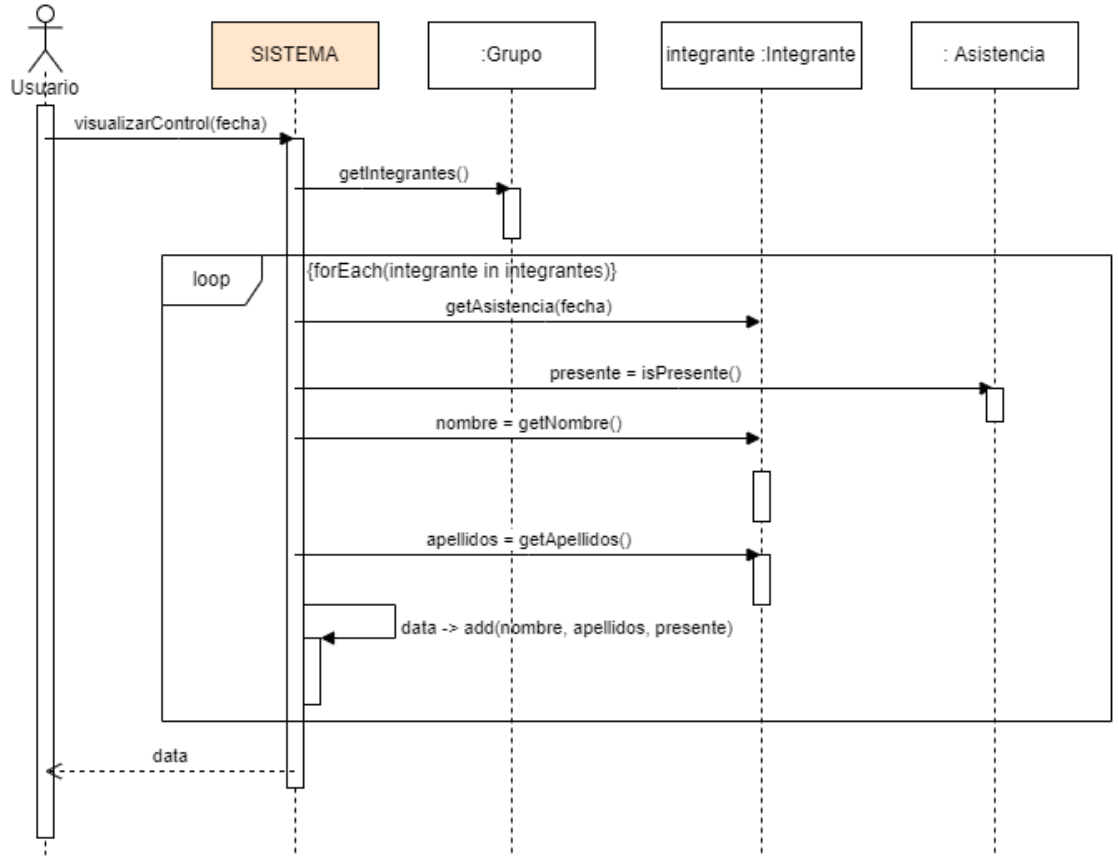
5.3.9. Caso de uso: Realizar control de asistencia



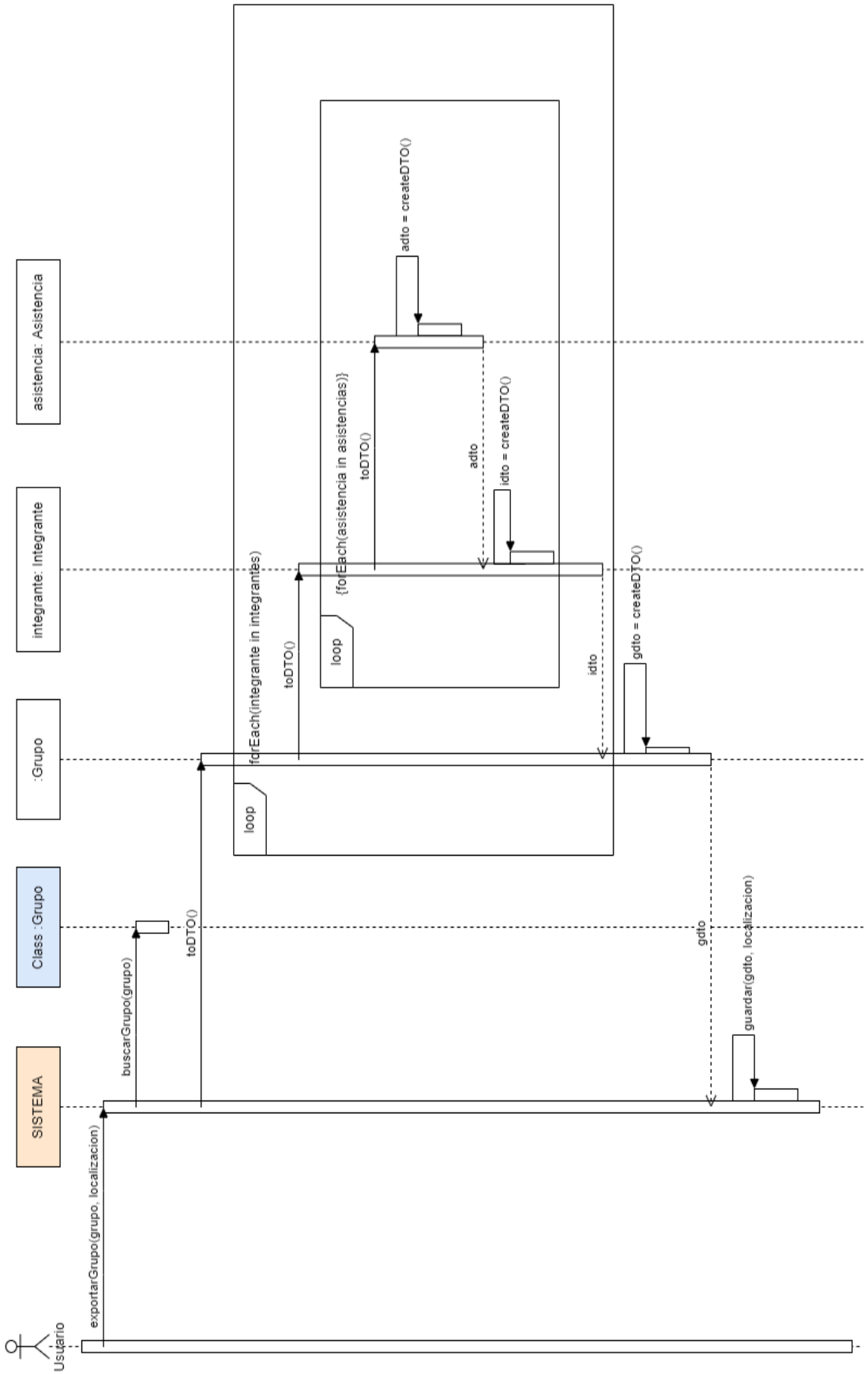
5.3.10. Caso de uso: Modificar control de asistencia



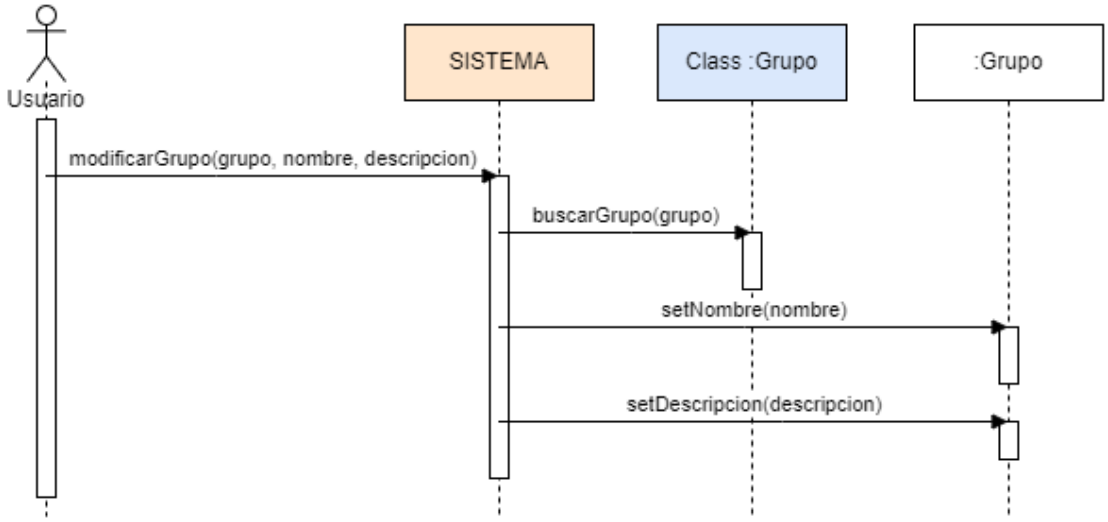
5.3.11. Caso de uso: Visualizar control de asistencia



5.3.12. Caso de uso: Exportar Grupo



5.3.13. Caso de uso: Modificar Grupo

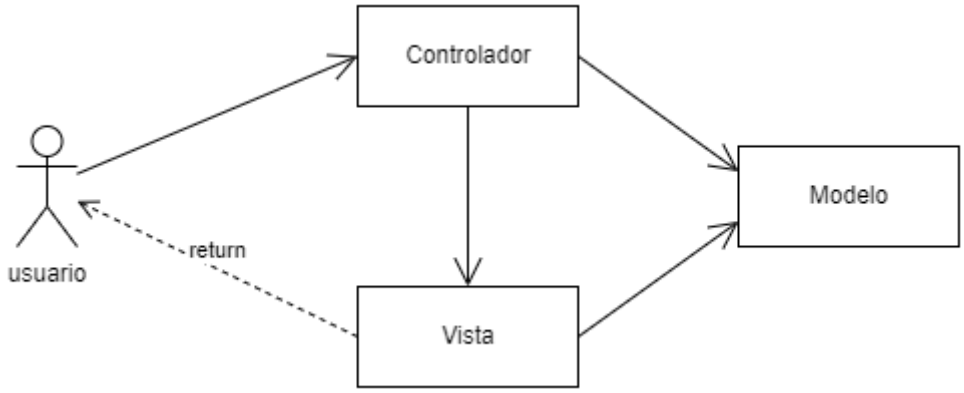


6. Arquitectura

Finalizado el análisis de requisitos, modelo de dominio y casos de uso, podemos finalmente hablar de la arquitectura que vamos a utilizar para construir la aplicación.

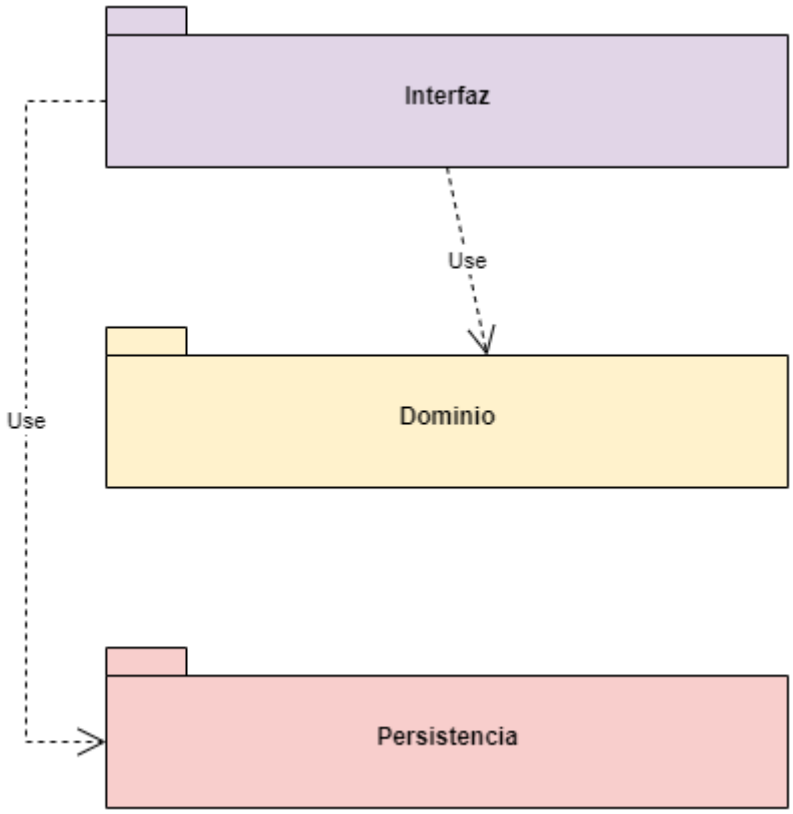
En este caso, este apartado va a ser bastante corto porque la arquitectura en este caso es bastante sencilla de definir. Recordamos que el [RNF-002] alude a una aplicación Android. Las aplicaciones de este sistema ya tienen una predisposición a emplear una arquitectura MCV (Modelo-Vista-Controlador). En el caso particular de Android, las vistas se componen de archivos XML, denominados *layouts*, que definen los diferentes componentes que presenta la vista, su disposición, etc. Los controladores se implementan usando clases Java que implementan las superclases *Activity* o *Fragment*. En nuestro caso, utilizaremos ambas implementaciones de controlador para definir una mejor distinción entre los diferentes grupos funcionales (entraremos más en detalle en el siguiente apartado).

A continuación, mostramos un pequeño esquema que explica gráficamente el comportamiento de este patrón arquitectónico:



Además de este patrón, aplicaremos un segundo patrón que define la jerarquía de responsabilidades que tiene cada componente. El patrón en cuestión será una arquitectura por capas. Concretamente, contaremos con tres capas: una capa de interfaz -que contendrá los controladores en nuestro caso-, una capa de dominio -será el que contendrá la implementación del modelo de dominio de la aplicación- y una última capa que será la de persistencia -encargada del almacenamiento y recuperación de datos persistentes-.

En nuestro caso, solo tendremos tres capas, donde la capa superior solo se comunica con las capas inferiores. No seguiremos un patrón completamente estricto ya que podemos aprovechar la capa de persistencia desde el controlador de la interfaz directamente para importar y exportar datos.



Finalmente, en cuanto al patrón de acceso a datos, usaremos el patrón DAO-DTO (Data Access Object - Data Transfer Object). Los DAOs serán las clases encargadas de acceder a la persistencia y usaremos los Objetos JSON como DTOs.

7. Diseño

[TODO]

Anexo

A. Glosario

- **El/este proyecto:** el desarrollo de este proyecto software en particular, incluyendo tanto la documentación formal presente en este documento, el manual de usuario que se incluirá al final del desarrollo y el propio aplicativo software que surgirá como producto final del proceso de desarrollo.
- **La aplicación:** el aplicativo para sistema Android que surgirá como producto final de este desarrollo. En palabras menos formales, el programa ejecutable y los componentes que lo forman.
- **El aplicativo:** mismo significado que la aplicación, descrito encima de este término.
- **El sistema:** mismo significado que la aplicación, descrito dos términos más arriba.
- **El usuario:** persona que utiliza la aplicación.
- **Modelo mental:** conjunto de ideas que uno tiene acerca de las entidades que le rodean. Cuando una persona oye la palabra "libro" en una conversación, a su mente acudirá una imagen de lo que entiende por "libro", permitiéndole entender el significado de la palabra. Esta imagen mental es una de millones que conforman lo que llamamos "modelo mental".
- **El Dominio:** es un conjunto abstracto de entidades que forman parte del sistema que tienen su equivalente en el modelo mental. Volviendo al ejemplo del libro utilizado para definir el modelo mental, la entidad que representa un libro tendrá, al igual que tiene un representante en el modelo mental, un representante en el dominio de una aplicación como puede ser el catálogo de una biblioteca. El "libro" de la aplicación de la biblioteca es diferente al "libro" del modelo mental de la persona que usa la aplicación, pero ambas entidades hacen referencia al mismo libro físico, a pesar de que todos podemos afirmar que el libro físico, la idea mental del libro y el libro del dominio son tres entidades diferentes.
- **Grupo:** componente del dominio que representa a un conjunto de integrantes. Su equivalente en el modelo mental sería el de un grupo de personas de los que es responsable el usuario y sobre el que desea llevar a cabo un control de asistencia.
- **Integrante:** componente del dominio que forma parte de un grupo. Su equivalente en el modelo mental sería el de una persona. El usuario está interesado en conocer si esta persona asistió o no a un evento organizado para el grupo al que pertenece.
- **Asistencia:** componente del dominio que representa a la asistencia de un integrante. Su equivalente en el modelo mental sería el de la anotación de "presente" o "ausente" en el momento de pasar lista a una persona concreta.

- **Grupo importable:** término utilizado para referirse a una representación de un grupo en un lenguaje de persistencia cuyo formato respeta los requisitos no funcionales [RNF-103], [RNF-104], [RNF-105] y [RNF-107].
- **Lenguaje de persistencia:** término utilizado para definir un lenguaje de marcado utilizado para la memoria persistente. Tal y como se describe en los requisitos no funcionales, estos lenguajes de marcado son -concretamente- XML, CSV y JSON.

B. Contenido del fichero *crazyjaeger_rollcall_group.dtd*

TODO

C. Contenido del fichero *example.new_group.txt*

TODO

D. Contenido del fichero *example.new_group.cvs*

TODO

E. Contenido del fichero *example.new_group.xml*

TODO

F. Contenido del fichero *example.new_group.json*

TODO

G. Contenido del fichero *example.group.cvs*

TODO

H. Contenido del fichero *example.group.xml*

TODO

I. Contenido del fichero *example.group.json*

TODO

J. Contenido del fichero *example.group_index.json*

[TODO]

K. Contenido del fichero *example.grp_group.json*

[TODO]